Failure prediction: what to do with unpredicted failures ?

Mohamed Slim Bouguerra*, Ana Gainaru* and Franck Cappello*
*INRIA UIUC and ANL Joint Laboratory for Petascale Computing

***Abstract*—As large parallel systems increase in size and complexity, failures are inevitable and exhibit complex space and time dynamics. Several key results have demonstrated that recent advances in event log analysis can provide precise failure prediction. The state of the art in failure prediction provides a ratio of correctly identified failures to the number of all predicted failures of over 90% and able to discover around 50% of all failures in a system. However, large parts of failures are not predicted and are considered as false negative alerts. Therefore, developing efficient fault tolerance strategies to tolerate failures requires a good perception and understanding of failure prediction characteristics. To understand the properties of false negative alerts, we conducted a statistical analysis of the probability distribution of such alerts and their impact on fault tolerance techniques. specifically we studied failures logs from different HPC production systems. We show that (i) the false negative distribution has the same nature as the failure distribution (ii) After adding failure prediction, we were able to infer statistical models that describe the inter-arrival time between false negative alerts and hence current fault tolerance can be applied to these systems. Moreover, we show that the current failures traces have a high correlation between the failure inter-arrival time that can be used to improve the failure prediction mechanism. Another important result is that checkpoint intervals can still be computed from an existing first-order formula.**

I. Introduction and background

The development and improvement of fault-tolerance mechanisms need realistic models in order to handle the failure occurrences in large-scale distributed systems. Traditional models have investigated failures in high performance computing systems (HPC) at much smaller scale, and often under the assumption of independence between failures. However, more recent studies have shown evidence that time patterns, correlations, and other time-varying behavior exist in the occurrence of failures.

With the introduction of failure prediction, a new direction of study has been opened. A complement to the classic preventive checkpoint-restart approach is failure avoidance, by which the occurrence of a fault is predicted and proactive measures are taken. This approach requires a reliable prediction system to anticipate failures and their corresponding locations. In general, two types of predictions are possible for HPC systems: (1) state prediction, where algorithms estimate the state of each node, and (2) failure prediction, where algorithms provide information about when and where failures will occur in the near future. The first type uses the states to decide whether a job can be scheduled on a specified node [30]. In this paper we use the second type. Specifically, we apply the ELSA tool [11] on historic logs generated by different HPC systems in order to record the predictions and conduct whetherstatistical analysis.

Before starting the analysis, we define some parameters and notations concerning failure prediction. A true positive alert is represented by a correct prediction that ends up with an actual failure in the system. False negative alerts represent actual failures that were not predicted by our method. False positives are predicted failures that did not actually happen in the system. Precision can be seen as a measure of fidelity; it represents the proportion of correct found anomalies to all identified anomalies. Mathematically the precision is given by $\frac{\#true\ positive}{\#true\ positive + \#false\ positive}$. Recall is computed as the ratio of correct identifications to all the existing failures in the log representing a measure of completeness; it is given by $\frac{\#true\ positive}{\#true\ positive + \#false\ negative}$.

Failure prediction techniques are based on the observation that an error propagation chain exists between the initial fault in the system and its corresponding events [27]. ELSA's approach assumes that faults generate a number of errors that could be observable at the system level, represented by notifications in the log files. The model used by ELSA improves other prediction techniques by considering the fact that different failures have different distributions and create different symptoms in the system. Current state-of-the-art approaches in failure prediction for HPC systems are based on data-mining algorithms and do not distinguish between the behavior of different failures. ELSA was shown to overcome the limitations of these approaches and was successfully applied on different systems, predicting approximately 45% of failures with a precision of 95%.

To benefit from failure prediction, we have to deal with several new challenges. One of the challenges is how to handle the 55% of the failures that are not visible by the prediction module. Also, false positives pose additional overheads on fault tolerance techniques. Thus, discovering and modeling the arrival of the unpredicted failures and true positive alerts influence the choice of the optimal fault tolerance strategy. Important objective addressed in this work is to compute the new checkpoint interval dedicated to the 55% of the failures. To this end, we first analyze the original distribution of failure arrival times and we use it to model the distribution of the unpredicted failure. Both distributions are used to describe the fundamental parameters that influence the tradeoff between the overhead due to fault tolerance actions and the amount of lost work due to failures in the presence of a prediction module. We show that a direct use case of this model is to compute the checkpoint interval dedicated to unpredicted failures. This study were conducted on a variety of systems recently in production at two US national laboratories. The results show that failure prediction mechanisms are able to detect the nonrandomness and correlation.

The rest of this paper is organized as follows. In Section II we review current related work and highlight our contributions. Section III presents the data we analyze throughout the rest of the paper. In Section IV we introduce the prediction method and its results on the proposed datasets. In Section V we develop our mathematical model for predicted and non predicted failure inter-arrival time. In Section VI, we present the experimental and simulation-based evaluations of our analytical model. We conclude this paper in Section VII with a brief summary and look at future work.

II. Related work

The work proposed in this paper differs from the existing work in two ways: what is measured and point is what is modeled. In terms of measurements, we consider the interval of time between false negative alerts versus the interval of time between failures.

Previous studies [29], [14], [15] focused on characterizing failures in several different distributed systems. Both [28] and [13] analyzed different HPC systems and found that Weibull distribution provides a good fit for the failure characteristic. The work in [34] has the same conclusion but also reveals the impact of filtering the failures that do not affect jobs on the distribution parameters and numerical characteristics. An example of an observation made in the paper is the way the probability of job interruption is related with existing historical records of application-related interruptions. The relationship between workload and failure rate in large-scale systems was also studied in [26] with similar conclusions.

The impact of failures and checkpointing protocols on application efficiency are studied in [18]. The authors analyze Blue Gene/L logs for six months by using temporal filtering and then by studying the job interruptions through simulation. A study that uses real job logs is presented in [34], where the analysis focuses on the impact of failures on the jobs in the whole system and not on specific failures. In [17], the authors do a similar study but with the focus on characteristics of troubleshooting from large-scale storage systems.

Most of these studies assume that failures occur independently, or they disregard the correlation of the time interval between failures. In this work we show that time-correlated failures have significant implications for proactive fault tolerance strategies.

Concerning the second point, we model in our work the distribution of the unpredicted failures versus the study done by related work by analyzing all failure occurrences.

A number of failure prediction methods, are presented in the literature. In [6], the authors introduce the concept of dynamic metalearning, where the prediction engine switches between different methods depending on different rules. A similar but less complex method can be seen in [9], where the authors extract rules for a fixed time window and generate association rules between fatal and non fatal events. Another approach for analyzing the logs is given in [7], where usage and failure logs are investigated by extracting past and future failure distributions for each failure instance. Based on these features, different decision tree classifiers are used in order to predict failures within a fixed time window.

In [25], the authors present middleware between the application and different analysis modules. One of the possible usages for their system is to facilitate proactive fault tolerance mechanisms such as preemptive job migration for fault predictors and other decision-making engines that rely on distributed failure information. However, they offer the middleware without having any of the system implemented. In our previous work [1], we implemented a hybrid fault tolerance protocol by combining preventive and proactive checkpointing strategies and modeled the impact of such a protocol on future exascale systems. The distributions of failures detected by a prediction system and of non detected failures greatly influence the overhead that fault tolerance techniques put on HPC applications [1].

In this paper, we focus on modeling the failures in the presence of a prediction model. To the best of our knowledge, all the existing studies focus on modeling all failures that occur in a system or failures that affect job executions and do not deal with how prediction changes their characteristics.

III. Data

The proposed analysis is based on event logs taken from different HPC. We analyzed 22 high-performance computing systems that have been in production use at Los Alamos National Laboratory (LANL) and a Blue Gene/L machine from Lawrence Livermore National Laboratories (LLNL). Traces represent different periods of time of production of high performance computing system logs. Table I reports the number of nodes, time intervals, and number of events that each trace contains. The BlueGene/L has 128K PowerPC 440 700 MHz processors, which are organized into 128 midplanes. A midplane is the granularity of job allocation. A midplane contains 16 node cards that represent the compute nodes, 4 I/O cards, and 24 midplane switches. Events are logged through the Machine Monitoring and Control System (CMCS) and stored in a DB2 database engine. The granularity of the Reliability Availability and Serviceability reporting system (RAS) is less than 1 second. Traces for systems at LANL represent events generated by a set of diverse systems. Systems vary widely in size, with different numbers of nodes ranging from 1 to 1,024 and different numbers of processors per node ranging from 4 to 6,152. The hardware architecture used by each system presents a large variety of different processor types and memory models. The system we are focusing on in our study, system 20, consists of 256 nodes and 1k processors, each with 16 GB of memory. We chose these systems because they have been analyzed in many papers and have been accepted as representative of HPC production systems. Most workloads for all systems are large-scale, long-running 3D scientific simulations. The main characteristic of these applications is the interleaving of hour long periods of CPU computation with minute periods of I/O for checkpointing and with periods of scientific visualization.

A. Failure identification

A preliminary step before focusing on the prediction method is to analyze the failures of each system. All traces offer information about events generated in the syslog for each system without specifically identifying the failures.

TABLE I. INFORMATION ABOUT SYSTEM TRACES

System	Nodes	Time Interval	MTBF (h)	Number of Events
Blue Gene/L	128K PowerPC 440 processors	June 2005 - January 2006	24.4	4,747,963
LANL systems	Different architectures	December 1996 - November 2005	From 13 to 125	433,490 per system

The 22 systems that were in production at LANL complement the system logs with manually annotated failure logs. System administrators at LANL registered the time for every failure in the systems and gave a brief explanation for all. By correlating the failures in this annotated log and the events in the syslog, we were able to decide which event represents a failure.

Blue Gene/L system (BG/L) offers only syslog traces and no information about failures except the severity. We use the filtering methods and anomaly detection techniques described in [10] to isolate events that have the potential of being failures after which we manually investigate all event types. Each failure event in our study does not necessarily correspond to a unique physical failure in the system hardware or software. Some of our reported failures, especially different types of failures that are reported in proximity to each other, may represent the same failure encountered by subsequent jobs. However, severity field can be misleading in many cases. For example in [31], while analyzing the Blue Gene/L system, the authors found that the severity field of the log messages performed poorly as an alert indicator and that after incorporating this metric into the failure detection algorithm the result would produce a false positive rate of 59%. The problem arises because of the lack of information about the exact duration of each failure. It is an extensive and difficult process to pinpoint the real root cause of each failure event or its actual duration. If this is not done at production time by system administrators registering information about every failure at the moment the failure occurs, it is almost impossible to get this information postmortem. For this reason, in this paper we do not isolate the impact of job execution on the failure pattern. Failure identification is an important step in the analysis of a system since both the prediction and the failure distribution analysis rely on an accurate set of failure events.

B. Failure statistics

We analyzed all the failures and extracted their statistical properties for each of the systems in Tables I and II. In this study we consider the annotated failure information for the LANL systems and the filtered set of failures for the Blue Gene machine. For the LANL systems, the system administrators divided the failure types into six categories: facilities, hardware, network, software, human, and unknown [29]. We clustered the Blue Gene/L events into similar categories in order to have a unified view of all the systems. Table II shows the percentage of each type of failure for each system. Human error has no representation for the Blue Gene/L system because traces do not give context information about the failures and so the actual root cause is unknown. Taking the analysis a step further, we investigate in the next section how the prediction results change when the analysis is done on different categories of failures.

TABLE II. PERCENTAGE OF ERROR TYPES

LANL		Blue Gene/L	
Facilities	2%	Node cards	16%
Hardware	62%	Midplane switch	4%
Human Error	<1%	Memory	22%
Network Error	2%	Network	17%
Software	23%	APP_IO	25%

IV. FAILURE PREDICTION

In the following subsections we present the methodology used for preprocessing the log files in order to predict failures. An overview of the methodology is presented in [10]. Our trace analysis comprises two phases: the offline phase, where we identify failures and construct the correlation chains between non faulty events and failures and between failures themselves, and the online phase, where the chains are used for prediction. The offline phase uses the first three to five months for each of the traces; and the rest time is used for online prediction.

A. Preprocessing

In the preprocessing step, we use the Hierarchical Event Log Organizer (HELO [12]) on the raw logs, which generates a list of message templates that represent frequently occurring messages with similar syntactic patterns. These templates represent regular expressions that describe different events in a system. We consider each template as representing an event type, and we analyze them separately by extracting a signal for each of them and characterizing their behavior and the correlations between them [10].

In general, filtering is used to reduce the size of the analyzed dataset without losing the log's characteristics. Contrary to the general use, our filtering method, described in [10], focuses on removing the normal behavior of the system in order to highlight the outliers. This allows us to isolate events related to failures and facilitates the extraction of event patterns by the prediction modules.

B. Failure prediction

For the prediction phase we use two separate methods for which we analyze their characteristics in the next sections. The first method uses a statistical-based method and is used only as a complement to the main prediction module. We describe this technique and study how it influences the results in the experiments section.

The main prediction method extracts correlations between non fatal and fatal events by using signal analysis to characterize events and data-mining algorithms to find patterns between them regardless of their behavior. The result of the offline phase is chains of events that end with a failure that might affect an application running on different nodes of a system. The online phase monitors the incoming stream of events and decides when to trigger a prediction. Also, modules in this phase update the correlations and the characteristics of events

TABLE III. BGL: BREAKDOWN OF PREDICTION RESULTS

Failure Type	Number Failures	Number Events	Recall
Node Cards	6	96	61%
Memory	251	8206	45%
Network	941	1055	15%
APP_IO	1019	1723	62%
Midplane switches	52	166	41%

TABLE IV. LANL: BREAKDOWN OF PREDICTION RESULTS

	Facilities	Hardware	Human Error
Precision	89.2%	93.8%	80.8%
Recall	38%	45.1%	9.2%
	Network Error	Software	Unknown
Precision	91.2%	93.7%	91.6%
Recall	42.8%	41.1%	23.4%

behavior to reflect the state of the entire system at different moments. Details about this process can be found in [11].

For this paper, the output of the prediction method is a list of predictions, each containing a timestamp and a system node. The list contains both correct predictions and false positives. Recall and precision values for all systems can be seen in Tables I, III, and IV.

We divide the predicted events into different categories. The results for system 20 from LANL are presented in Table IV by following the same failure clustering as in the previous section. For Blue Gene/L we focused our attention on the most frequent failure categories. The results are presented in Table III. In general, there is a large difference of coverage between different types of failures, which indicates that certain failure types appear in patterns and correlations more than do others. Depending on the resources an application might use and hence which parts of the system are more stressed and prone to failures, the overheads and benefits of preventive checkpointing techniques might vary.

V. STATISTICAL MODELING

All the decision makers of the existing fault tolerance strategies and mechanisms are based on the mathematical stochastic models that describe the inter-failures arrival time and the failures prediction alert times. Technically the decision makers uses the inter-failures arrival time models as input to manage the tradeoff between the overhead due to fault tolerance actions such as checkpointing, migration, or replication versus the amount of lost work due to failures.

The objective of this section is to discover the mathematical model that can be used to describe these two fundamentals inputs. We focus mainly on the two stochastic processes.

The first is the classical stochastic model that describes the inter-arrival time between failures. This model is used, for instance, to compute the optimal interval between checkpoints [5], [33], [2] or to schedule jobs in order to minimize completion time and maximize the reliability in same time [16].

The second stochastic process considered in this work concerns the interval of time that separates two unpredicted failures (two false negative alerts). To the best of our knowledge this is the first work that model this stochastic process. In fact recent proactive fault tolerance strategies [1], [21] combine proactive fault tolerance action and preventive tolerance actions to handle failures. In those works the distribution of the false negative alerts is used as input to schedule preventive checkpoints.

The most common approach for describing this kind of stochastic process is to model the time between failures or false negative alerts by a sequence of continuous and positive random variables denoted by $U_1, U_2, \cdots, U_i$ for failures and $Y_1, Y_2, \cdots, Y_i$ for false negative alerts. U_i or Y_i is the interval interval of time between the $i-1$ and i failure or false negative alert. Figure 1 shows an example of the relation between the sequence Y_i and the sequence U_i. Suppose now that failures 1, 3, and 4 are correctly predicted, in this case the false negative intervals are given by $Y_1 = U_1 + U_2$ and $Y_2 = U_3 + U_4 + U_5$. Recall that U_i denotes the interval of time between failure i and failure $i-1$. Based on this model, we infer the probability

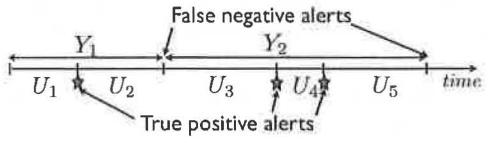


Fig. 1. Failure and false negative intervals

distribution of U and Y from the empirical data. The data used in this work reports failures of each node. Indeed, building a stochastic model for each node is too expensive in terms of computation and irrelevant for the existing fault tolerance decision-makers. In fact one needs a model that describes the failure arrival that concerns the entire cluster used to run the application. In this work we consider the cluster as one unit. Thus, formally, the failure time of a system is the first failure that occurs on any system node. However, recall that this model is valid only if the intervals of time between failures or false negative alerts are independent and identically distributed. In the next section we investigate whether the raw data validates this hypothesis of true randomness.

A. Randomness testing

As a preliminary phase and before distribution fitting we run several tests of randomness to identify whether the failures extracted have a truly random behavior. The objective of this phase is to decide whether the data set is from a random process and does not exhibit any trends of periodicity, autocorrelation, or non-stationarity. We note that it is statistically irrelevant to fit probability distribution to nonrandom data, since such data don't fill the basic assumption of all standard statistical tests, namely is that the sample is obtained by a random procedure. Also building a statistical model for this nonrandom data is difficult given the change in terms of the trend over time. In this section the objective is twofold.

1) We check the randomness hypothesis of failure intervals and identify whether it is possible to fit the data to classic probability distribution functions. Also, by studying the randomness of the false negatives alerts, we investigate the impact of the prediction process on the randomness of the data.
2) We investigate the relationship between the randomness of the failure intervals and the efficiency of

the failure prediction mechanism. This will determine whether a dataset with a non truly random behavior has a better recall than does a a truly random dataset.

1) Methodology: In the literature, randomness tests are classified into two categories. The first category, called *parametric tests*, is often used when we have information about the distribution of the data. Since we don't have any assumption about the distribution we need randomness tests from the *non-parametric tests* category. This category contains three well-known non-parametric tests. First we have the runs test, also called the Wald and Wolfowitz test [32], where each interval of time is compared to the mean. This test verifies that intervals are mutually independent. The second test called run up/down [3] is also similar to the runs test, it is designed to capture the trend of the data set by comparing each interval to the previous one. The third test is an autocorrelation test [19], and it is used to discover repeated patterns that differ only by a lag in time. We note that when the autocorrelation is used to detect nonrandomness, usually only the first lag autocorrelation is of interest.

As there is no perfect method to judge randomness, we run the three tests over all the data. The runs test and the up/down test return one value, called a probability value, denoted by a p-value. This value is used to either reject the null hypothesis about the randomness of the data if the p-value is smaller than or equal to the significant threshold, or confirm that the data is truly random if the p-value is greater than the significant threshold. As pointed in [3] and [32], for the first and second tests a p-value of 5% is a strong evidence to reject the null hypothesis about the randomness of data. For the autocorrelation we consider a confidence interval of 95%. Thus, if the value of the autocorrelation test is out of this confidence interval, the sample contains a high correlation of order 1, which implies the nonrandomness of the data.

2) Randomness test results: Table V reports the p-values for the runs test and the up/down test concerning both real failures and false negative alerts. We note that all the statistical tests are based on the normal distribution assumptions. Thus, we have to make sure that the input samples are enough. According to Casella and Berge [4] a sample should contain more than 30 entries. Therefore, both system 22 and system 24 are excluded from the study since they contain respectively 17 and 23 failures.

Highlighted cells represent the case where the data fail to pass the randomness test. This first result shows that more than the half of the failure traces data concerning the failures intervals fail to pass the test of randomness. This implies that fitting classical probability distribution to this data is irrelevant, since the sequence of interval is not random. Hence all the existing works that use a failure distribution to make decisions about checkpointing or scheduling jobs cannot be used in this case.

Most of the traces concerning the interval of time separating the unpredicted failures pass both runs and up/down tests except for four cluster. Thus, the failures with nonrandom pattern are predicted and removed from the traces. Intuitively, this means that the failure prediction mechanism catches most of the failures with nonrandom occurrences. Thanks to

TABLE V. RANDOMNESS TESTS P-VALUES

System Name	Failures			False negative		
	# Lines	Runs Test	Up/Down Test	# Lines	Runs Test	Up/Down Test
Blue Gene/L	235	0.11	0.17	129	0.70	0.97
LANL Sys 2	1951	0.01	0.17	1172	0.01	0.86
LANL Sys 3	294	0.08	0.73	158	0.36	0.92
LANL Sys 4	298	0.75	0.42	163	0.15	0.83
LANL Sys 5	304	0.51	0.95	158	0.83	0.59
LANL Sys 6	63	1.00	0.88	32	0.69	1.00
LANL Sys 8	436	0.30	0.03	270	0.69	0.48
LANL Sys 9	279	0.01	0.23	172	0.01	0.10
LANL Sys 10	234	0.22	0.72	122	0.07	0.13
LANL Sys 11	266	0.01	0.56	154	0.11	0.63
LANL Sys 12	255	0.01	0.19	154	0.01	0.02
LANL Sys 13	194	0.04	0.74	123	0.80	0.53
LANL Sys 14	120	0.06	0.36	75	0.49	0.17
LANL Sys 15	53	0.01	0.87	32	0.50	0.51
LANL Sys 16	245	0.04	0.98	159	0.62	0.97
LANL Sys 18	3917	0.01	0.01	2195	0.66	0.74
LANL Sys 19	3235	0.03	0.54	1785	0.08	0.86
LANL Sys 20	2400	0.01	0.14	1310	0.01	0.85
LANL Sys 21	105	0.02	0.01	76	0.39	0.96
LANL Sys 22	17	not	enough	lines		
LANL Sys 23	226	0.32	0.41	129	0.15	0.55
LANL Sys 24	23	not	enough	lines		

failure prediction we can use statistical fitting techniques to mathematically model the false negative alerts.
To investigate this intuitive result further and to study the impact of the nonrandomness on the prediction process, we show in Figure 2 the autocorrelation values using the first lag. Horizontal lines in Figure 2 represent the 95% confidence interval. The autocorrelation test reveals also that most of the data concerning the failures intervals exceed the confidence interval. This result confirms that failure traces are not truly random and show a high correlation. This also confirms previous studies [14], [28] that report a high correlation between failures. As with the for runs and up/down tests, after using the failures prediction, the false negative traces are truly random except for few a systems.

Figure 2 shows that systems 9, 10, 11, 12, 13, 14, and 15 present a high correlation. This can be explained by the fact that those clusters have the same hardware type, denoted by the type F in [28]. Among this set of systems the correlation is different because the in system size. One would expect that the bigger the size, the higher the correlation; but this is not true. In fact the smallest clusters 9, 10, and 12 have the highest correlation.

Bars in Figure 4 represent the recall ratio for the different computing systems. This figure points out another important finding. The recall ratio is related to the amount of correlations that the traces contain. For instance, for system 21 which has a smaller recall, Figure 2 indicates that it also has the smallest amount of correlation. We have the same observation for system 10; again the recall is high, and it has the highest amount of correlation. Figure 4 shows that we can still get high recall even if the trace does not contain a high correlation of order 1, for example, system 6 has the highest recall and a small correlation value. Considering that a trace is random only if it passes the three tests, only 6 of 20 computing systems with a truly random failure sequence can be used to infer statistical models to describe mathematically the arrival time of failures. We note that for the systems without true randomness, no failure distribution could be estimated. Thus, one cannot use fault tolerance strategies such as those in [5], [33], [1], [21] to compute the optimal interval between checkpoints. One solution could be to change the scale and consider failures on the node level, in order to insulate the nodes with high correlation. Also, failures with truly random behavior lead to false negative samples that are truly randomly distributed. The important finding is that, thanks to the failure prediction mechanism that catches the failures with periodic patterns and correlation, we can infer statistical models that describe the inter-arrival time between false negative alerts for 15 of 20 computing systems.

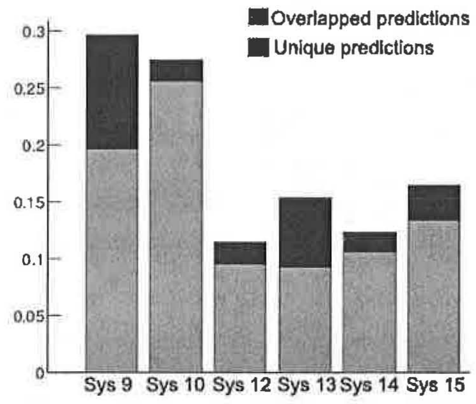


Fig. 3. Recall for the statistical-based prediction method

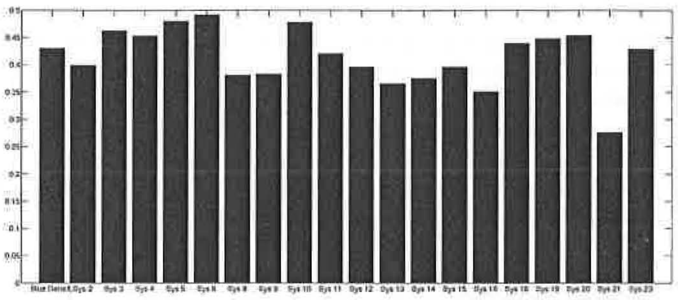


Fig. 4. Recall ratio for various systems

The lack of correlation between Figures 2 and 4 influenced us to implement a statistical-based prediction method, similar to the one from [6] that uses the failure distribution for each of the analyzed systems. statistical-based methods emphasize discovering probabilistic characteristics among failure events and then using the obtained characteristics for failure prediction. Our method basically involves two phases. In the offline phase it obtains and verifies statistical characteristics of failures, for example temporal correlations from the training data, and in the online phase it produces a warning if statistical patterns are observed.

The results are closely related to the amount of correlation between inter-arrival time of failures in the LANL traces. Figure 3 presents how statistical-based methods can complement our correlation-based prediction and indicates the impact on the recall value for the LANL systems.

Clearly, system 9 presents a strong correlation between failures, and as a consequence the prediction technique offers high recall values. In some cases, systems that do not have a strong correlation are misinterpreted by the prediction technique and thus create a large number of false positives, decreasing the overall precision. This is the case of system 8, where the statistical-based method discovers several weak false correlations that are later used for false predictions. To avoid this problem, we decided to keep only very strong predictions and filter the rest. Figure 3 presents only the systems that have strong correlations, and as a result the gain in recall outperforms the loss in precision.

On closer examination, we observed that the failure predicted by the statistical-based prediction method overlaps with some of our original predictions. Moreover, the introduction of the new predictor does not change the shape of the recall from Figure 4 and the improvement in recall does not exceed 15%. This observation led us to an important conclusion, specifically that the statistical-based prediction technique is not universal and provides different results depending on the system. Techniques like the one in [6] can be used only for systems with high correlations for the failure inter-arrival time, and their results are specific to the system they analyze.

B. Distribution fitting

The principle behind fitting distributions to data is to find the type of distribution (normal, lognormal, gamma, beta, etc) and the value of the parameters (mean, variance, etc) that give the highest probability of producing the observed data. The objective here is to infer a mathematical model that can be used to describe formally the inter-arrival time between failures and false negative alerts. We then investigate the relationship between the failures distribution without prediction and the probability distribution of the false negative alerts. Only traces with truly random behaviors are relevant to be used to find a good probability distribution that is a good match to the empirical distribution.

1) Fitting methodology: Different methods are available to fit the empirical data to probability distribution functions. The most common methodology is: first select a set of candidate distributions then estimate the values of distribution parameters based on the empirical distribution. The best fit with the most likely similarity is kept. Many distributions could be used as input candidate in step 1. In this work we conduct the distribution fitting process using the commonly used distribution functions to model failures in HPC systems [15], [14], [28], namely, exponential, Weibull, log-normal, normal, and gamma. In the second step we compute the best parameter values for each candidate distribution. Specifically, in step 2 we look for the maximum likelihood estimates (MLE) [23] that are the most likely fit to the empirical data. We choose this method rather than using the moment matching method since this method is sensitive to outliers [8]. Technically, MLE aims to maximize the logarithm of the likelihood function that corresponds to the closet distance between the empirical distribution and samples resulting from distribution with certain parameters. We use the negative log likelihood value produced by the MLE to rank the different distributions. This still does not mean that this distribution is a good model for the empirical data. Thus we check also the goodness of fit between the data sample and synthetic sample. The literature describes dozens of goodness-of-fit tests, but only a handful are used in practice. We use the Kolmogorov-Smirnov [24] test and the standard probability-probability plot (PP-plot) as a visual method. The Kolmogorov-Smirnov test checks that the sample comes from the best-fitted distribution against the alternative that it does

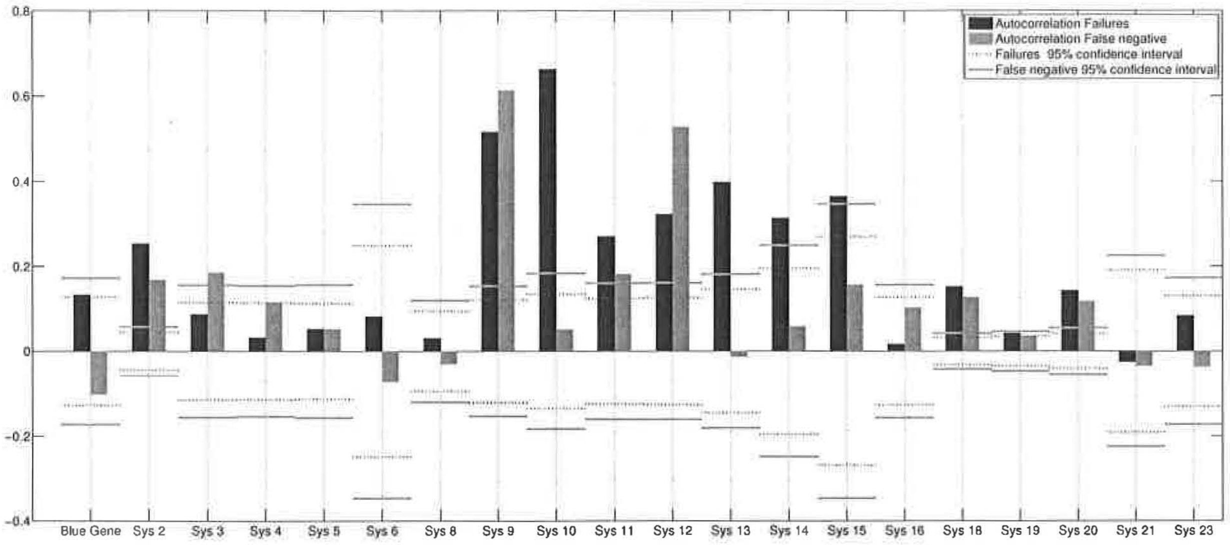


Fig. 2. First lag autocorrelation coefficients

not come from the late distribution. Also the test rejects the true randomness hypothesis at the 5% significance level.

2) Fitting results: We first investigate the statistical model for clusters with a true random behavior for both failures and false negative alerts. Then we consider traces that fail to pass the randomness test for failures intervals and pass the randomness test for false negative intervals. We use the second time scale to estimate the failure distribution parameter. Table VI reports the fitting results concerning the first set of data where both intervals are truly random. As we can see, not all the probability distributions are present, but exponential and Weibull are the closet fit to the available data.

We note that the parameter μ is the mean in seconds for the exponential distribution. For the Weibull parameter a denotes the scale and b the shape parameter. The exponential distribution is a good fit for the data with a coefficient of variation (CV) [20] close to 1. This is an expected result since the CV of the exponential distribution is equal to 1, hence if the sample is taken from exponential random variables, its CV should be close to 1 as well. For the data with a high variation, Weibull is the best fit. We notice also that systems have either a constant failure rate for the exponential case or a strictly decreasing failure rate, since all the shape parameters are lower than 1 for the Weibull case.

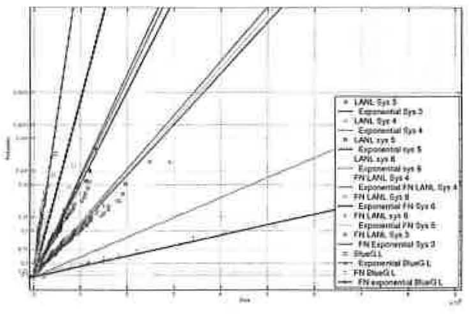
The second outcome from this study is the relationship between the initial failure distribution and the false negative distribution function. As can been seen in Table VI the best-fitted distribution for the data concerning the false negative alerts is the same distribution for the failure intervals with different parameters. Hence, intuitively we can say that the failure prediction process does not change the initial distribution and affects only the scale parameters of the initial distribution. Also as reported in Table VII, for the case where the distribution is exponential, the ratio between the initial parameter μ_u and the false negative parameter μ_y is given by $\mu_y/\mu_u \approx 1-r$, where $r = 0.45$ is the recall. As with Weibull, we have approximately the same shape parameter for both distributions ($b_u = b_y$), and the scale parameters verify $a_u/a_y \approx 1 - r$. This means that the failure prediction mechanism acts as a scaling filter affecting only the time scale. Therefore we can avoid the fitting process and estimate the distribution of the false negative alerts using the recall and the initial failure distribution. We note that the failure prediction process does not have an impact on the variability of the data. As we can see in Table VI the coefficient of variation (CV) is almost the same for both data.

The Kolmogorov-Smirnov test values (denoted by KS) in Table VI indicate that all the fitting successfully passes the test of goodness. To assess the fitting results by a visual method, we report in Figures 5(a) and 5(b) the PP-plots. Clusters are grouped based on the scale and the kind of failure distribution. Figure 5(a) reports the PP-plot for the exponential type distributions. As can be seen the PP-plot confirms that the exponential distribution presents a good visual fitting. We have the same observation in Figure 5(b) reporting the PP-plot for Weibull-type distribution.

Figure 5(a) and Table VI show that systems 3, 4, and 5 have almost the same failure distribution and the same recall, which leads to the same false negative distribution as well. We note that systems 3, 4, and 5 have exactly the same architecture; more precisely, they have the same processor type and number [22]. Moreover, Schroeder and Gibsion [28] report that more than 50% of the failures experienced by this set are CPU-related failures due to a design flaw. This implies that the failures observed on those three different systems are independent from the workload profiles or the applications executed on these clusters. Cluster number 6 is also from the same family, but it has only 32 nodes whereas the clusters in the other set are composed of 128 nodes. We observe that the ratio between the two failure rates is approximately 4.6, which corresponds to the ratio between node numbers. This observation confirms that the failure rate

TABLE VI. STATISTICAL OF FITTING ALL RANDOM PROCESS (FITTING PARAMETERS SCALE ARE IN SECONDS)

System Name	Failures				False Negative				$\frac{\mu_u}{\mu_y} \approx 1 - r$ $\frac{a_u}{a_y} = 1 - r$
	Mean	CV	Best Fit	KS	Mean	CV	Best Fit	KS	
Blue Gene/L	1040.5	0.92	exponential $\mu_u = 62431.3$	0.10	1888.1	1.10	exponential $\mu_y = 113289$	0.79	0.55
LANL Sys 3	3595.1	1.1	exponential $\mu_u = 215705$	0.98	6559.0	1.1	exponential $\mu_y = 393538$	0.70	0.54
LANL Sys 4	3409.1	1.1	exponential $\mu_u = 204544$	0.77	6187.0	1.1	exponential $\mu_y = 371218$	0.99	0.54
LANL Sys 5	3294.5	1.1	exponential $\mu_u = 197671$	0.95	6377.9	1.2	exponential $\mu_y = 382671$	0.35	0.51
LANL Sys 6	16796.7	0.9	exponential $\mu_u = 1007800$	0.81	31878.2	1.1	exponential $\mu_y = 1912690$	0.99	0.54
LANL Sys 23	9288.2	1.3	Weibull $a_u = 509380$ $b_u = 0.846905$	0.97	16272.3	1.2	Weibull $a_y = 895274$ $b_y = 0.851258$	0.98	0.56


(a) PP-plot systems: BlueG/L, 3, 4, 5 and 6

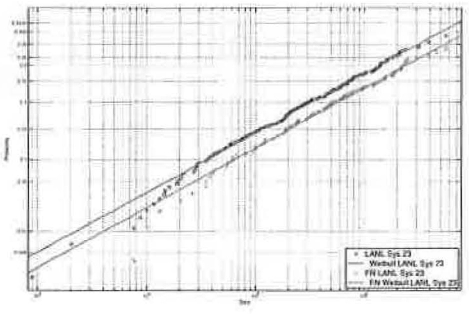

(b) PP-plot system 23

Fig. 5. PP-plot and CDF for exponential type distribution (a) and Weibull-type distribution (b).

is linearly proportional to the size of the system when the hardware is the same.

Table VII reports the fitting results for the set of systems with nonrandom failures and random false negative alerts. As can be seen, the set of best fit distributions is different. Contrary to the previous case, here the exponential distribution, the Weibull distribution, and the lognormal are the best candidates to model the inter-arrival time of false negative alerts. To double check the fitting results, we report the PP-plot in Figures 6 and 7 for the different systems.

TABLE VII. STATISTICAL FITTING FOR RANDOM FALSE NEGATIVE

System Name	False Negative			
	Mean (min)	CV	Best Fit	KS
LANL Sys 8	7859.6	1.4	Weibull a = 401499 b = 0.767798	0.74
LANL Sys 10	8247.0	3.6	Weibull a = 318087 b = 0.647838	0.29
LANL Sys 11	6353.5	3.0	Weibull a = 232647 b = 0.609348	0.61
LANL Sys 13	8164.3	3.9	lognormal μ = 11.5257 σ = 1.87004	0.14
LANL Sys 14	11351.0	2.5	Weibull a = 391931 b = 0.559039	0.77
LANL Sys 15	12136.7	1.2	exponential μ = 728203	0.17
LANL Sys 16	3430.6	1.3	Weibull a = 182624 b = 0.810939	0.69
LANL Sys 18	818.6	1.5	lognormal μ = 10.1123 σ = 1.28677	0.37
LANL Sys 19	863.6	1.4	exponential μ = 51816.8	0.18
LANL Sys 21	1986.9	2.3	lognormal μ = 10.6382 σ = 1.46402	0.85

One question that may be raised is whether one can model failure arrival times of the entire LANL computing system as one unit. The answer is no, since some samples in the data comes from process that are not non truly random and the only way to analyze the failure sets is to break the system into smaller units. This result shows the benefit of understanding temporal correlations and shows how one can exploit them for failure prediction. Moreover, it provides a new opportunity to design smart fault tolerance strategies and enhance the conventional ones.

VI. USE CASE

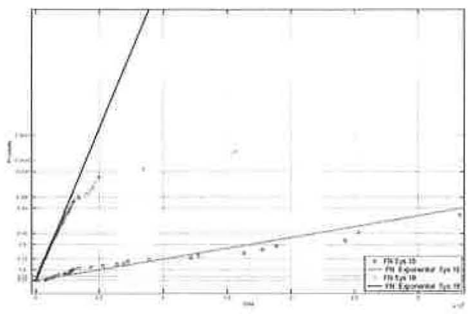
Many use cases can be developed to show the impact of such models. In this paper we present a case where checkpoint and migration are coupled in order to reduce the overall wasted time. The proposed fault tolerance strategy is to use migration as a proactive action when a prediction alert is available. Arguably proactive migration alone cannot systematically avoid re-executing the application from scratch if failures are not perfectly predicted. Therefore, failure prediction and proactive migration should be combined with periodic checkpointing. However, coupling failure prediction with proactive migration and periodic checkpointing is not trivial. In order to provide significant benefits, proactive and preventive actions scheduling strategies should be chosen carefully. Basically, we have to compute the optimal interval between periodic checkpoints and to decide online whether it is worthwhile to perform proactive migration in light of the failure prediction information.

A. Preventive checkpoint interval

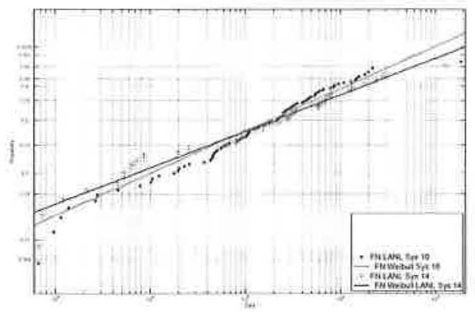
The preventive action is to perform periodic checkpointing of the current application state. This action is performed in a constant amount of time, c. We introduce τ to represent the units of useful work between two consecutive preventive checkpoints. In this work we consider that a unit of work corresponds to a unit of time as well. We use the classical Young's formula $\tau = \sqrt{2c\mu}$, where μ is the mean time between unpredicted failures.

B. Proactive migration

The proactive action is considered efficient if and only if the alert is a true positive alert. In order to handle false positive alerts and to minimize the wasted time due to them, the decision to perform or not the proactive action is taken

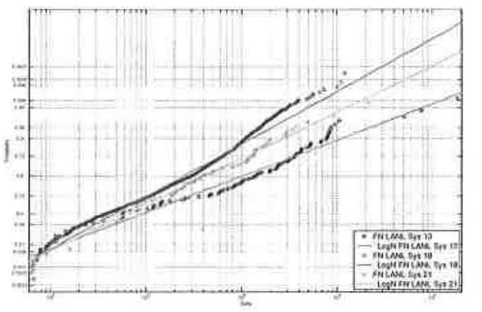


(a) PP-plot systems 15 and 19

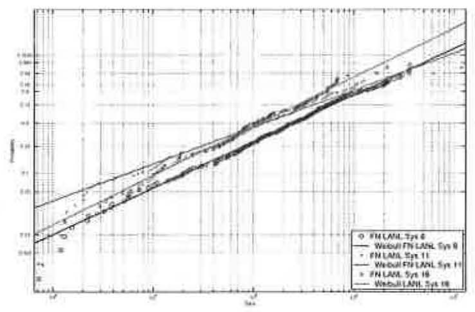


(b) PP-plot systems 10 and 14

Fig. 6. PP-plot and CDF for exponential-type distribution (a) and Weibull-type distribution (b).



(a) PP-plot systems 13, 18 and 21



(b) PP-plot systems 8, 11 and 16

Fig. 7. PP-plot and CDF for lognormal type distribution (a) and Weibull-type distribution (b).

online according to the progress of the application since the last checkpoint and the precision of the alert versus the migration cost Let t_a denote the elapsed time since the last checkpoint; p is the precision of the prediction mechanism, and m is migration cost. As shown in [1], the proactive action is performed if the inequality $\overline{p}m/p \leq t_a$ holds; otherwise it will be ignored.

C. *results*

We present in this section the overall improvement thanks to the proposed combination. We investigate using an event-based simulator the percentage of useful work that an application can reach in presence of failures. To this end we use the traces of failures, false negative and true positive of the system 19 at LANL. We reply an execution considering several configurations of checkpoint cost between 10 and 40 minutes. Migration is considered as constant cost at 1 minute. As indicated in Table VII the false negative distribution is exponential. Bars in Figure 8 show the overall percentage of useful work that an application can reach. we compare the proposed combination versus a classical execution using the Daly [5] optimal interval with proactive migration. In order to compute the optimal interval for Daly model we use the mean time between failures in the traces. This figure shows that the application efficiency is improved by more than 5% with a recall of 45%. This confirms the expected theoretical improvement shown in our previous work [1].

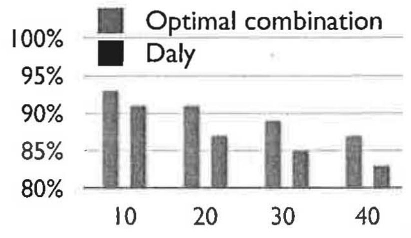


Fig. 8. Overall system useful time

VII. Conclusion

A major challenge facing parallel applications on large scale HPC systems is failures. The situation is predicted to be even worst for exascale platforms. For this reason, understanding the temporal correlations of failures and exploiting them for smart checkpointing and scheduling decisions are critical tasks. Our study highlights that most of the available failure traces are not random and hence are suitable for use as empirical data for probability fitting. This result suggests that developers of failure analysis algorithms should focus on understanding the temporal correlations of failures. Moreover, we point out that the failure prediction mechanism is a good tool to identify and remove the nonrandomness and correlation. Another important, unexpected, result concerns the impact of the relation between the initial failure distribution and the false negative distribution. In this work we show that the failure prediction mechanism acts as a scale function and affects only the scale parameter. Hence, when the initial distribution is available, based on it, we estimate the distribution of the false negative alerts, which avoids the overhead of the fitting process. We also show that the peak of correlation on the initial traces has an important impact on the prediction results, specifically on the recall value. This influences the design of some adaptive fault tolerance strategies. For example, a simple scheduling policy could be to stop scheduling large parallel jobs during failure peaks. Moreover, one can devise adaptive fault-tolerance mechanisms that adjust the policies based on the information related to peaks. For example, an adaptive fault-tolerance mechanism can migrate the computation at the beginning of a predicted peak. We plan to analyze more deeply the set of systems with a high correlation like system 2 or 20, and isolate sources of nonrandomness. Doing so can lead to higher precision in terms of prediction. Another direction of future research is to investigate whether a cross-correlation of different time scales has an impact on the prediction mechanism.

Acknowledgments

This work was supported by the Advanced Scientific Computing Research Program, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

References

[1] Mohamed Slim Bouguerra, Ana Gainaru, Franck Cappello, Leonardo Bautista Gomez, Naoya Maruyama, and Satoshi Matsuoka. Improving the computing efficiency of hpc systems using a combination of proactive and preventive checkpointing. In *Proceedings of IEEE IPDPS 2013*. IEEE press, 2013.

[2] Mohamed Slim Bouguerra, Derrick Kondo, and Denis Trystram. On the scheduling of checkpoints in Desktop grids. In *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid 2011)*, CCGRID '11, pages 305–313, NewPort Beach, CA, USA, May 2011. IEEE Computer Society.

[3] J.V. Bradley. *Distribution-free statistical tests*. Prentice-Hall Englewood Cliffs, NJ, 1968.

[4] George Casella and Roger L Berger. Statistical inference. *Duxbury Press*, 2001.

[5] J. T. Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Computer Systems*, 22(3):303–312, 2006.

[6] J. Gu et al. Dynamic meta-learning for failure prediction in large-scale systems: A case study. In *International Conference on Parallel Processing*, pages 157–164. IEEE press, 2008.

[7] N. Nakka et al. Predicting node failure in high performance computing systems from failure and usage logs. In *IEEE Workshop on Dependable Parallel, Distributed and Network-Centric Systems*. IEEE press, 2011.

[8] Dror G Feitelson. Workload modeling for performance evaluation. In *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures*, pages 114–141. Springer-Verlag, 2002.

[9] Xiaoyu Fu, Rui Ren, Jianfeng Zhan, Wei Zhou, Zhen Jia, and Gang Lu. Logmaster: Mining event correlations in logs of large-scale cluster systems. In *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*, pages 71–80, Oct.

[10] Ana Gainaru, Franck Cappello, and William Kramer. Taming of the shrew: Modeling the normal and faulty behavior of large-scale hpc systems. In *Proceedings of IEEE IPDPS 2012*. IEEE press, 2012.

[11] Ana Gainaru, Franck Cappello, Marc Snir, and William Kramer. Fault prediction under the microscope: A closer look into hpc systems. In *Proceedings of 2012 International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE press, 2012.

[12] Ana Gainaru, Franck Cappello, Stefan Trausan-Matu, and Bill Kramer. Event log mining tool for large scale hpc systems. In *Proceedings of the 17th international conference on Parallel processing - Volume Part I*, Euro-Par'11, pages 52–64, Berlin, Heidelberg, 2011. Springer-Verlag.

[13] T. Hacker, F. Romero, and C. Carothers. An analysis of clustered failures on large supercomputing systems. *Journal of Parallel and Distributed Computing*, 69:652–665, 2009.

[14] E. Heien, D. Kondo, A. Gainaru, D. LaPine, B. Kramer, and F. Cappello. Modeling and tolerating heterogeneous failures in large parallel systems. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 45. ACM, 2011.

[15] B. Javadi, D. Kondo, JM. Vincent, and D.P. Anderson. Discovering statistical models of availability in large distributed systems: An empirical study of seti@home. *IEEE Transactions on Parallel and Distributed Systems*, 22(11):1896 –1903, 2010.

[16] E. Jeannot, E. Saule, and D. Trystram. Optimizing performance and reliability on heterogeneous parallel systems: Approximation algorithms and heuristics. *Journal of Parallel and Distributed Computing*, 2012.

[17] W. Jiang and T. et al Zhou. Understanding customer problem troubleshooting from storage system logs. *7th USENIX Conference on File and Storage Technologies*, 2009.

[18] W. Jones, J. Daly, and N. DeBardeleben. Impact of suboptimal checkpoint intervals on application efficiency in computational clusters. *HPDC*, 2010.

[19] J.D. Knoke. Testing for randomness against autocorrelation: Alternative tests. *Biometrika*, 64(3):523–529, 1977.

[20] Lambert H Koopmans, Donald B Owen, and JI Rosenblatt. Confidence intervals for the coefficient of variation for the normal and log normal distributions. *Biometrika*, 51(1/2):25–32, 1964.

[21] Zhiling Lan and Yawei Li. Adaptive fault management of parallel applications for high-performance computing. *Computers, IEEE Transactions on*, 57(12):1647–1660, 2008.

[22] LANL. Failures traces. http://institutes.lanl.gov/data/fdata/, 2008.

[23] Erich Leo Lehmann and George Casella. *Theory of point estimation*, volume 31. Springer, 1998.

[24] Jr. Massey and J. Frank. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[25] Raghunath Rajachandrasekar, Xavier Besseron, and Dhabaleswar K Panda. Monitoring and predicting hardware failures in hpc clusters with ftb-ipmi. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1136–1143. IEEE, 2012.

[26] R. Sahoo, A. Sivasubramanium, M. Squillante, and Y. Zhang. Failure data analysis of a large-scale heterogeneous server environment. *DSN*, 2004.

[27] Felix Salfner, Maren Lenk, and Miroslaw Malek. A survey of online failure prediction methods. *ACM Computing Surveys*, 42:1–42, 2010.

[28] B. Schroeder and G.A. Gibson. A large-scale study of failures in high-performance computing systems. *IEEE Transactions on Dependable and Secure Computing*, 7(4):337–351, 2006.

[29] Bianca Schroeder and Garth A. Gibson. A large-scale study of failures in high-performance computing systems. In *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks*, pages 249–258, Washington, DC, USA, 2006. IEEE Computer Society.

[30] J. Stearley, R. Ballance, and L. Bauman. A State-Machine Approach to Disambiguating Supercomputer Event Logs. *MAD*, 2:155–192, 2012.

[31] Narate Taerat, Nichamon Naksinehaboon, Clayton Chandler, James Elliott, Chokchai Leangsuksun, George Ostrouchov, Stephen L Scott, and Christian Engelmann. Blue gene/l log analysis and time to interrupt estimation. In *Availability, Reliability and Security, 2009. ARES'09. International Conference on*, pages 173–180. IEEE, 2009.

[32] A. Wald and J. Wolfowitz. On a test whether two samples are from the same population. *The Annals of Mathematical Statistics*, pages 147–162, 1940.

[33] J. W. Young. A first order approximation to the optimum checkpoint interval. *Commun. ACM*, 17(9):530–531, 1974.

[34] Z. Zheng and L. et al Yu. Co-analysis of ras log and job log on blue gene/p. *Proceedings of the 2011 IEEE International Parallel and Distributed Processing Symposium*, pages 840–851, 2011.